CMPS 111 - Spring 2002
Midterm Exam
May 1, 2002

Name:___Answer Key_____ID:_____

This is a closed note, closed book exam. There are 20 multiple choice questions and 6 short answer questions. Plan your time accordingly.

Part I: Multiple Choice - Indicate your answer to the left of each question (2 points each)

1. What are the two functions of an operating system:
   a. Resource management and resource manipulation
C b. Resource sharing and resource management
   c. Resource abstraction and resource sharing

2. Multiprogramming is:
   a. Multiple processes running at different times
B b. Multiple processes residing in memory and running more or less at the same time
   c. Multiple processes running at exactly the same time

3. Which has the fastest access times:
   a. Cache
B b. Registers
   c. Main memory

4. 4. Which are all system calls:
   a. chmod(), chdir(), read(), main()
C b. waitpid(), smdir(), mount(), open()
   c. fork(), exit(), lseek(), kill()

5. Which is true about processes and threads
   a. Threads in a process share the same stack
B b. Threads in a process share the same file descriptors
   c. Threads in a process share the same register values

6. A race condition is when:
   a. Multiple processes are all trying to finish first
C b. A process runs too slow
   c. The correctness of the code depends upon the timing of the execution

7. Mutual exclusion is when:
   a. A set of processes are prevented from running because higher priority processes keep getting in ahead of them
C b. A set of processes is waiting for an event that only another process in the set can cause
   c. A set of processes are prevented from simultaneously accessing a shared data structure

8. Semaphores, Locks and Condition Variables, and Monitors support two distinct operations
   a. Parallelization and event notification
B b. Synchronization and event notification
   c. Synchronization and parallelization

9. Which is true:
   a. Round-robin has the highest turnaround time
C b. Shortest remaining time next is the fairest scheduling algorithm
   c. Priority-based scheduling is the most general

Not Graded (B)

10. The difference between preemptive and non-preemptive scheduling is:
   a. Whether or not a ready process can be involuntarily terminated
   b. Whether or not a running process is involuntarily removed from the ready state
   c. Whether or not a blocked process can have its resources involuntarily taken away from it

11. The four conditions that must hold in order for deadlock to occur are:
A a. Mutual exclusion, hold-and-wait, circular wait, and no preemption
   b. No preemption, no starvation, circular wait, and mutual exclusion
   c. Hold-and-wait, circular wait, no starvation and mutual exclusion

12. Four strategies for dealing with deadlock are:
B a. Conservative resource allocation through the banker's algorithm, spooling, two-phase locking, and mutual exclusion
   b. Do nothing, kill one of the processes, preemptible resources, and spooling
   c. Two-phase locking, resource ordering, resource discovery, and do nothing

13. Which is false:
   a. Memory compaction is time consuming
C b. Degree of multiprogramming refers to the amount of time a process spends waiting for I/O
   c. Swapping can result in external fragmentation

14. The main advantage of bitmapped memory management is:
A a. Its simplicity
   b. Its space efficiency
   c. Its speed

15. The main advantage of implementing free list memory management in the holes themselves is:
B a. Its speed
   b. Its space efficiency
   c. Its simplicity

16. The main advantage of paging is:
    a. That it doesn't require any extra hardware support
C   b. The efficiency with which data can be brought into memory
    c. That only the current working set of the process need actually be in memory at any one time.

17. The main advantage of multilevel page tables is that they:
    a. Use page table memory efficiently
A   b. Reduce the complexity of the paging system
    c. Speed up page table references

18. Page replacement algorithms:
    a. Least Recently Used is an approximation of Not Recently Used
C   b. Optimal page replacement is usually the best choice, unless you know the reference stream beforehand
    c. Second Chance is a minor variation of FIFO

19. Which are both Stack algorithms
    a. Optimal and LRU
A   b. LRU and FIFO
    c. FIFO and Optimal

Not graded (B)

20. Which is not true:
    a. Large pages can lead to wasted memory in the form of internal fragmentation
    b. Small pages can lead to wasted memory due to excessive page table size
    c. Large pages can lead to wasted memory in the form of internal fragmentation

Part II: Short answers (10 points each)

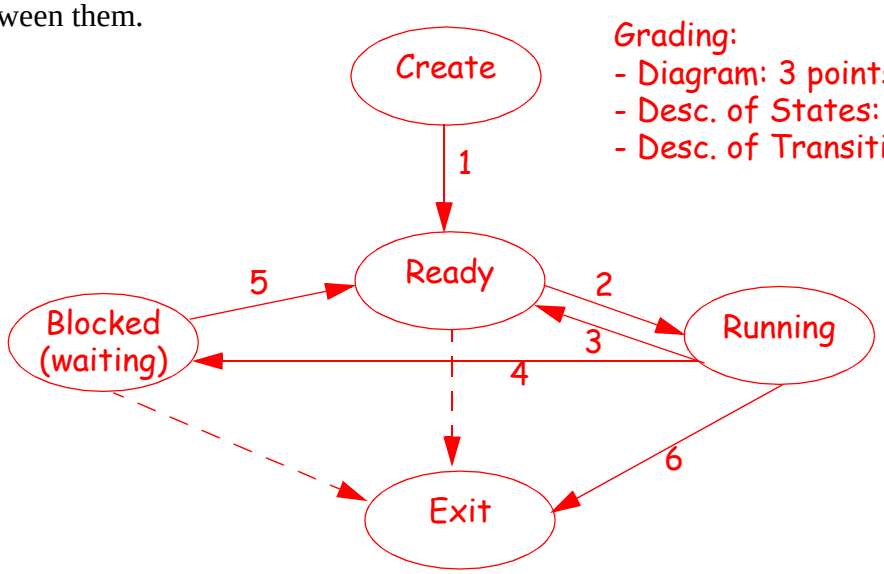
1. Explain the purpose of system calls in an operating system and give an example, explaining how your explanation applies to it.

System calls allow user processes to request service from the kernel. In particular, they allow the user to ask the kernel to do something for the user process that is not allowed to do for itself because of protection or sharing concerns. For example, user processes are allowed to access some files but not others, and have read access to some file and read/write access to others. Thus the write() system call allows a user process to ask the OS to write some data to a file. Before performing the operation, the OS verifies that the user has write access for that file.

In addition, the system calls abstract away the details of the hardware and the lower level software, providing a clean, uniform, and portable application programming interface to the user processes.

Grading:
- Requesting service from the kernel: 3 points
- Protection/sharing: 2 points
- Abstraction: 2 points
- Example: 3 points

2. Draw a process state diagram and very briefly describe each state and the transitions that occur between them.



Grading:
- Diagram: 3 points
- Desc. of States: 3 points
- Desc. of Transitions: 4 points

There are five states that a process can be in:
- Create: A process in this state is being created, but it not yet ready to run.
- Ready: A process in this state is ready to run, i.e. it has everything it needs to run except the CPU.
- Running: A process in this state is actively executing. It has everything it needs to run, including the CPU. Only one process may be in this state at any one time.
- Blocked: A process in this state is waiting for something that it needs before it can continue executing. It may be waiting for a resource that is currently held by another process, an event that another process or the system will cause, or just for a certain time.
- Exit: A process in this state has exited but is kept around in a zombie state until the parent process does a wait() operation to verify that the process has finished and to read the exit status.

There are six transitions that a process can take between states:
1. Create->Ready: The process has been created and is ready to execute
2. Ready->Running: The scheduler decides to execute the process
3. Running->Ready: The scheduler decides to stop running the process and run something else instead. It may also take this transition voluntarily by calling yield() in systems that support this system call
4. Running->Blocked: The process needs something (resource, data, other I/O, event) before it can continue executing
5. Running->Exit: The process terminates. Termination can be voluntary or involuntary, but the process always has to be running before it can exit
6. Blocked->Ready: The data or even the process is waiting for has arrived or occurred

3. Show the schedules that would result given the following processes with the scheduling algorithms below:

Process	Arrival Time	Processing Time	Priority
0	0	10	4
1	0	5	7
2	10	10	3
3	10	1	1

a) First-Come First-Serve (2 points)

Notation: (Time,Process,Duration)
Schedule: (0,0,10), (10,1,5), (15,2,10), (25,3,1)

b) Round Robin (time quantum = 5) (3 points)

Schedule: (0,0,5), (5,1,5), (10,2,5), (15,3,1), (16,0,5), (21,2,5),

c) Priority (2 points)

Schedule: (0,0,10), (10,3,1), (11,2,10), (21,1,5)

d) Shortest Remaining Time Next (3 points)

Schedule: (0,1,5), (5,0,5), (10,3,1), (11,0,5), (16,2,10)

4. Unix, MacOS, and Windows all employ do-nothing deadlock management. List (and briefly explain) three reasons why the designers might have decided to manage deadlock this way. Hint: it might be helpful to think of the costs of the deadlock management algorithms that we have examined.

Possible reasons:
1. Simplest "solution" to implement - This strategy requires the least coding, effort, money, etc. to implement. All of the other strategies require doing something
2. More efficient - It doesn't require any runtime checking. All of the other strategies require some checking to make sure that resources are requested in the right order, the system is in a safe state, deadlock has not occurred, etc. Furthermore, resources can be aggressively allocated since we don't have to worry about safety.
3. Unimportant processing - The processing done on a PC is (was) unimportant, and so deadlocks will not cause any major problems. Users can kill processes or reboot the system if necessary.
4. Deadlock is rare - Deadlock doesn't happen all that often in mature code, so the expense and computational overhead of implementing any other strategy is not worth it.

Grading:
- 1 point for writing anything
- 3 points per good reason and explanation

5. A computer has four page frames. The time of loading, time of last access, and the R (reference) and M (modify) bits for each page are as shown below (the times are in clock ticks):

Page	Loaded	Last Reference	R	M
0	126	280	1	0
1	230	265	0	01
2	140	270	0	0
3	110	285	1	1

a) Which page will NRU replace?

2

b) Which page will FIFO replace?

3

c) Which page will LRU replace?

1

d) Which page will Second Chance replace?

2

e) Which page will Optimal replace?

That depends on the future accesses, which we do not know.

Grading: 2 points each

6. Describe the purpose and basic operation of a generic paging system.

The purpose of paging is to allow for a higher degree of multiprogramming within a given amount of memory or to allow processes larger than will fit in main memory by efficiently simulating a system with a lot more memory. Without paging, each process requires memory for all of its code and data. Paging limits the amount of memory allocated to each process by only loading into memory the code and data that the processes are actively using. As a result, each process uses less memory, and more processes can be brought into memory at one time.

The pages for each process are managed by a page table that indicates whether or not each logical page of the process' address space is in memory or not. If a page is in memory, the page table indicates where in physical memory the page can be found. If the page is not in memory, then there is some indication of where on disk the page can be found. In addition, for pages that are in memory, the page table contains information about whether or not the page has changed since it was last loaded into memory or written to disk (dirty bit) and whether or not the page has been referenced since the reference bit was last cleared (reference bit).

Because it is expensive to manage page tables and address translation in software, much of this is managed by the hardware.

Grading:
- Purpose: 5 points
- Description: 5 points